Recuperatorio primer parcial de Laboratorio III

PARTE 1 (hasta un 5)

Crear una aplicación Web que permita realizar un ABM de distintos aliens.

Crear la siguiente jerarquía de clases en **Typescript** (en el namespace **Entidades**):

a. ***Ente***: cuadrante(cadena), edad(entero) y altura(flotante) como atributos. Un constructor que reciba tres parámetros. Un método, *ToString():string,* que retorne la representación de la clase en formato **cadena** (preparar la cadena para que, al juntarse con el método ToJSON, forme una cadena JSON válida).

b. ***Alien***, hereda de Ente, posee como atributos raza(cadena), planetaOrigen(cadena) y pathFoto(cadena). Un constructor para inicializar los atributos. Un método *ToJSON():JSON,* que retornará la representación del objeto en formato **JSON**. Se debe de reutilizar el método *ToString* de la clase Ente.

Crear en **TypeScript** la clase ***Manejadora*** (en el namespace **RecuperatorioPrimerParcial**) que posea los siguientes métodos y funcionalidades:

AgregarAlien. Tomará los distintos valores desde la página *index.html* (incluida la foto), creará un objeto de tipo **Alien**, que se enviará (por ***AJAX***) hacia ***"./BACKEND/adminstrar.php"***. En esta página se guardará al alien en el archivo ***"./BACKEND/alien.json"*** y la foto en ***"./BACKEND/fotos"***.

MostrarAliens. Recuperará (por ***AJAX***) todos los aliens del archivo .json y generará un listado dinámico (en el ***FRONTEND***) que mostrará toda la información de cada uno de los aliens (incluida la foto).

GuardarEnLocalStorage. Recuperará (por ***AJAX***) todos los aliens del archivo .json y los guarda en el LocalStorage, con la clave "aliens_local_storage".

VerificarExistencia. Verifica que el alien que se quiere agregar no exista. Para ello, comparará los cuadrantes y la raza de los aliens guardados en el LocalStorage. Si el alien existe, se mostrará (por **consola** y **alert**) lo acontecido. Caso contrario, agregará el nuevo alien y se actualizará el LocalStorage.

ObtenerAliensPorCuadrante. Recupera del LocalStorage todos los aliens y muestra, por consola, que cuadrante o cuadrantes poseen más aliens (y su cantidad) y que cuadrante o cuadrantes poseen menos aliens (y su cantidad).

PARTE 2 (hasta un 6)

Agregar una columna (Acciones) al listado de aliens que permita: **Eliminar** y **Modificar** al alien elegido. Para ello, agregue dos botones (input [type=button]) que invoquen a las funciones EliminarAlien y ModificarAlien, respectivamente.

En la clase Manejadora, agregar la interface IParte2:

EliminarAlien. Eliminará al alien del archivo (por ***AJAX***) y del LocalStorage. Recibe como parámetro al objeto ***JSON*** que se ha de eliminar. Pedir confirmación, mostrando cuadrante y raza, antes de eliminar. Refrescar el listado para visualizar los cambios.

ModificarAlien. Mostrará todos los datos del alien que recibe por parámetro (objeto ***JSON***), en el formulario, incluida la foto (mostrarla en ***"imgFoto"***). Permitirá modificar cualquier campo, a excepción del cuadrante, dejarlo como de solo lectura.
Modificar el método ***AgregarAlien*** para cambiar el caso de *"agregar"* a *"modificar"* y el texto del botón de *"Agregar"* a *"Modificar"*, según corresponda.
Refrescar el listado solo si se pudo modificar, caso contrario, informar (por alert y consola) de lo acontecido.
Modificará al alien del archivo y del LocalStorage.

PARTE 3

En la clase Manejadora, agregar la interface IParte3:

FiltrarAliensPorPlaneta. Mostrará (por ***AJAX***) un listado dinámico (en el ***FRONTEND***) de todos los aliens según el planeta seleccionado.

CargarPlanetasJSON. Cargará (por ***AJAX***) en el combo (cboPlaneta) con el contenido del archivo **"./BACKEND/planetas.json"**.

LimpiarForm. Vaciará todos los campos del formulario. Este método se invocará cada vez que se realice una acción sobre el formulario. El combo de planetas quedará seleccionado el valor "Kriptón".

AdministrarSpinner. Este método mostrará u ocultará el archivo **"./BACKEND/gif-load.gif"**, de acuerdo al parámetro booleano que recibe como parámetro. Aplicar la llamada de este método en cada acción que invoque a un AJAX.

- En el **index.html**, agregar un checkbox a cada input para que el listado solo muestre las columnas que están seleccionadas.

- Generar la pre visualización de la foto del alien antes de guardarla (tanto para el alta como para la modificación)

Crear en TypeScript las funciones necesarias para verificar que **todos** los campos de la página, **index.html**, sean enviados correctamente.

La función se llamará ***AdministrarValidaciones*** y será la encargada de invocar a otras funciones que verifiquen:

2- Campos no vacíos (todos).

a. ***ValidarCamposVacios(string): boolean***. Recibe como parámetro el valor del atributo id del campo a ser validado. Retorna true si no está vacío o false caso contrario.

3- Tipos correctos (tipo).

a. ***ValidarRaza(string, string[]): boolean***. Recibe como parámetro el valor a ser validado y los valores permitidos para las razas (Grises, Reptilianos, Draconianos, Kriptonianos, Melmacquianos). Retorna true si el valor pertenece a los tipos o false caso contrario.

4- Selección del código (código).

a. ***ValidarEdad(number): boolean***. Recibe como parámetro el valor de la edad a ser validada y retorna true si la misma es mayor o igual a 0 y menor a 133. False caso contrario.

Si algún campo no pasa la validación, se mostrará un asterisco (*****) al lado de dicho campo y no se permitirá el envío de la información. Si todos los campos pasan todas las validaciones, se enviará la información correspondiente.

Aplicarlo tanto para el alta como para la modificación de los aliens.

IMPORTANTE:

Se pueden bajar templates de internet o traer código hecho, pero en ningún caso se debe incluir código obsoleto o que no cumpla ninguna función dentro del parcial.

Toda la comunicación con el backend, se realizará con AJAX. El pasaje de datos, con JSON.

Los puntos se evaluarán de manera descendente y consecutiva.